METHOD OF COMPRESSING SOUNDS IN MOBILE TERMINALS

BACKGROUND OF THE INVENTION

FIELD OF THE INVENTION

<1> The present invention relates to a method of compressing bell sounds using recorded sounds and voice memos (hereinafter, referred to 'sounds') in mobile terminals and, more particularly, to a method of compressing sounds in mobile terminals, which compresses pulse code modulation (PCM) code generated by sampling sounds, using Lempel Ziv Welch (LZW) compression technique by applying a differential method.

DESCRIPTION OF THE PRIOR ART

<2> Generally, mobile terminals use Musical Instrument Digital Interface (MIDI) or recorded bell sounds in order to inform users of phone calls. MIDI bell sounds have been developing from existing mono-poly sounds to poly-poly sounds, and the recorded bell sounds use recorded music or voice to satisfy personal taste. Also, mobile terminals store voices so as to store details of the calling during on the line or to leave a memo during call waiting.

<3> Presently, a method of storing sounds including bell sounds and voice memos used in mobile terminals uses a method of storing sounds using Adaptive Differential Pulse Code Modulation (ADPCM) compression algorithm without using sounds coder/decoder (CODEC) for supporting high tone quality provided by mobile terminals. Such ADPCM compression algorithm can reduce a storage space by about half level, but it cannot resist a degradation of tone quality.

<4> In the existing method of storing sounds, voices are stored by transforming data

sampled into PCM using ADPCM. PCM algorithm has been disclosed in International Telecommunications Union-Telecommunication Standardization Sector (ITU-T) G.711 Recommendations and ADPCM algorithm has been disclosed in ITU-T G.721 Recommendations.

<5> The sounds storage method using the existing ADPCM described above has been improved, but it still has problems in that memories are excessively consumed and original sounds cannot be restored as they are because the method uses compression technique causing damage of source data.

SUMMARY OF THE INVENTION

<6> Accordingly, the present invention has been made to solve the above-mentioned problems occurring in the prior art, and an object of the present invention is to provide a method of compressing sounds, which increases compression efficiency by transforming input data to be suitable for LZW compression algorithm through applying differential method to PCM code generated by sampling sounds.

<7> In order to achieve at least the above objects, in whole or in parts, there is provided a method of compressing sounds in mobile terminals, including: initializing differential code corresponding to difference between adjacent PCM codes among PCM codes generated by sampling input sounds, in a dictionary table; sequentially reading PCM codes generated by sampling actually inputted input sounds, transforming the PCM codes into corresponding differential codes initialized in the dictionary table, and outputting the differential codes; and registering the outputted differential codes in a dictionary through dictionary generation algorithm.

<8> Preferably, in initializing the differential codes in the dictionary table, the

differential codes are 6-bit differential codes and the number of the differential codes is 64.

<9> Preferably, said sequentially reading the PCM codes, transforming the PCM codes into differential codes, and outputting the differential codes includes: producing differential code variables that are differences between previously read PCM code and presently read PCM code; and differently outputting the differential codes according to the produced differential code variables' values.

<10> Preferably, in said differently outputting the differential codes according to the produced differential code variables' values, if the produced differential code variables' values are in a certain range, the differential code variables are outputted as they are, on the other hand, if the produced differential code variables' values are not in the certain range, the differential code variables are transformed and outputted.

<11> Preferably, the certain range is a range that the produced differential code variables' values are equal to or more than 0 and less than 31.

<12> Preferably, if the produced differential code variables' values are not in the certain range, the differential code variables are classified again according to the values of differential code variables, and the corresponding differential code variables are transformed in different manners according to the classified values and outputted.

BRIEF DESCRIPTION OF THE DRAWINGS

The above and other objects, features and advantages of the present invention will be more apparent from the following detailed description taken in conjunction with the accompanying drawings, in which:

<13> Figure 1 is a flow chart of processes for sound compression in mobile terminals according to one preferred embodiment of the present invention;

<14> Figure 2 is a flow chart of differencing process illustrated in Figure 1;

<15> Figure 3 is a flow chart of dictionary generation function in the compressing process illustrated in Figure 1;

<16> Figure 4 illustrates output bit string of code word according to one preferred embodiment of the present invention;

<17> Figure 5 illustrates a structure of code word table of sound data according to one preferred embodiment of the present invention;

<18> Figure 6 illustrates the probability of PCM code of sampling sound data according to one preferred embodiment of the present invention; and

<19> Figure 7 illustrates the probability of differential code according to one preferred embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

<20> Hereinafter, preferred embodiments of the present invention will be described with reference to the accompanying drawings.

<21> Figure 1 is a flow chart of processes for sound compression in mobile terminals according to one preferred embodiment of the present invention, whose first process is to initialize 64 code words for 6-bit differential code in a dictionary table (S110).

<22> That is to say, as a result of analysis of PCM code obtained by sampling the recorded sounds in order to construct a code word required for sounds compression, the difference between neighboring PCM codes (the absolute value of a certain value obtained by subtracting one PCM code from neighboring PCM code) is less than 32, so

that only 64 code words that may be generated are stored in the dictionary table as differential codes, and code word variable (C1), which indicates the next code word to be registered, is initialized as the number of N5 (N5=65), which is initial dictionary entry number.

<23> Then, the stored PCM codes are sequentially read one by one (S120). The read PCM codes are processed with differencing so as to be mapped into 64 differential codes initialized in the dictionary table (S130). The differential codes after differencing are outputted to a function of compression (S140).

<24> According to the function of compression, the differential codes are compressed by using dictionary generation algorithm and the compressed code words are outputted and stored in a memory. At this time, the dictionary generation algorithm generates dictionary trees suitable for the differential codes.

<25> The steps (S120, S130 and S140) are repeated until all the PCM codes obtained by sampling are read (S150).

<26> Then, when the differencing and compressing of all the PCM codes are completed, a flush is finally conducted (S160). According to a storage method in a memory, data is stored by 8-bit or 16-bit. Since the number of bits of compressed data is variable, final data stored in a memory may not correspond to 8-bit or 16-bit. Thus, bits left are filled with 0 and the above process is called 'flush'.

<27> Respective processes of sound compression will be described in detail with reference to the drawings.

<28> Figure 2 is a flow chart of a process for differencing the PCM codes (S130).

<29> Referring to Figure 2, the corresponding differencing process is to transform 8-bit PCM code into 6-bit differential code, wherein PCM code previously read (old) is subtracted by PCM code presently read (cur) so as to obtain the differential value of

PCM code, and the subtracted value is stored in differential code variable (temp) (S201).

<30> Then, it is checked whether or not the value of differential code variable is within a range of initialized differential codes so as to map input sounds into 64 differential codes initialized in the dictionary table, using the differential code variable.

<31> For example, if the value of differential code variable ranges from 0 to 31 (31 is not included) (S202), the corresponding differential code variable is outputted as a differential code because the corresponding code variable is a differential code initialized in the dictionary table (S203). And, if the value ranges from –32 to 0 (0 is not included) (S204), 6-bit complement for 2 of differential code variable is outputted as differential code (S205).

<32> However, when the value of differential code variable exceeds the range of differential code initialized in the dictionary table, differential code variable goes through a certain processing. When the value of differential code variable ranges from –160 to –32 (-160 is not included) (S206), differential code 32 is outputted in order to indicate that the value of differential code variable is less than –32 and, then, an absolute value of the corresponding differential code variable divided by 2 is outputted as differential code (S207 and S208).

<33> When differential code variable ranges from 31 to 159 (159 is not included) (S209), differential code 31 is outputted in order to indicate that the corresponding differential code variable exceeds 31 and, then, a value of the corresponding differential code variable divided by 2 is outputted as differential code (S210 and S211).

<34> Figure 3 is a flow chart showing a step (S140) of compressing differential code transformed by the differencing process, using dictionary generation algorithm. A dictionary generated for compressing differential code can be previously generated upon fabricating mobile terminals or upon initially storing sounds.

<35> Referring to Figure 3, a case where a character string is not added to the dictionary is one where the character string exceeds the maximum number (N7) of character string (S301) or where the character string is previously registered in the dictionary (S302). The character string is allocated to a new code word C1 except upon the above two cases (S303).

<36> Then, new code word C1 increases by 1 so as to be allocated to the code word of character string to be generated next (S304). When the increased C1 is equal to or more than the number of code word (N2) (S305), the number of N5, initial dictionary entry number, is allocated to the C1 (S306). The steps (S304 to S306) are repeated until a node allocated to the C1 is a leaf node indicating last character of the character string in the dictionary tree, or a node that is not used (C1=NULL) (S307).

<37> Where the node allocated to the C1 is a leaf node or the node that is not used, C1 is deleted from the dictionary tree in order for new code word of the character string to be allocated (S308).

<38> When the compression has been completed through the above steps, generated code word is outputted and stored in a memory. To reduce the size of compressed code word, a process is conducted as follows. That is to say, in order to obtain accurate character string when decompressing the compressed code word, the corresponding code word is outputted as to satisfy the following equations.

【equation 1】

$$(C1+\mathrm{lim}) \leq 2^{\lceil \log_2(C1+1) \rceil} - 1$$

【equation 2】

lim=C3-C1-1

【equation 3】

$$C3 = 2^{\lceil \log_2(C1+1) \rceil}$$

<39> where C1 is the number of code word presently allocated, lim means a limit value capable of reducing bits, and $\lceil \log_2(C1+1) \rceil$ means minimum integer larger than $\log_2(C1+1)$. Accordingly, when code word is changed into bit string, if the code word is smaller than a predetermined limit value lim, it is outputted as $\lceil \log_2(C1+1) \rceil - 1$ bit, and if the code word is larger than a limit value, it is outputted as $\lceil \log_2(C1+1) \rceil$ bit.

<40> For example, as shown in Figure 4, since lim=(1024-750-1)=273, when C1 is 750. Code words ranging 0 to 273 upon being compressed are coded by 9 bits and outputted, and code words ranging 274 to 749 are coded by 10 bits after adding 274 to respective code words and are outputted.

<41> When decompressed, code word bits are read by 9 bits. If the read value is smaller than 274, the value itself is taken as a code word, on the other hand, if the value is larger than 274, code word bits are read again by 10 bits and a certain value subtracting 274 from the read value is taken as a code word.

<42> Figure 5 illustrates a structure of the dictionary table according to one preferred embodiment of the present invention. The code words ranging 0 to 63 are defined as differential code, code words ranging 64 to 127 as 7-bit coding area, code words ranging 128 to 255 as 8-bit coding area, and finally code words ranging 2048 to 4095 as 12-bit

coding area.

<43> In order to evaluate performance of the method of compressing sounds according to the present invention, compression algorithm is implemented using C language and tested. For sound data, actual human voice is recorded at 8000 samples per second (64Kbps) and used.

<44> Figure 6 illustrates the probability of PCM code of sampling sound data, and Figure 7 illustrates the probability of differential code, which records difference based on data from Figure 6.

<45> Compressibility of sounds according to the present invention is obtained by dividing the size of sounds data before compression by the size of sounds data after compression. With the result of this, samples 1 to 4 have compressibility of 3.00, 3.66, 3.35, and 2.5, respectively and average value of 3.13.

<46> As described above, the present invention can reduce the number of kinds of code word, a parameter which heightens performance of LZW compression algorithm, by applying differential method to PCM code generated by sampling sounds and can enhance sound compression efficiency by increasing the number of repeated character string.

<47> Although preferred embodiments of the present invention have been described for illustrative purposes, those skilled in the art will appreciate that various modifications, additions and substitutions are possible, without departing from the scope and spirit of the invention as disclosed in the accompanying claims.